

AC Automaton. Construct the AC search automaton for words `dar`, `radar`, `adam`, `a`.

Too many occurrences. Find an example input of patterns and text which has asymptotically more than linear number of occurrences. Specifically, for every n , show that there is input to the TEXT SEARCH problem such that $|T| + \sum_i |P_i|$ is $\Theta(n)$, and the number of occurrences is not $\mathcal{O}(n)$.

Naive jumps. Consider the simplified AC algorithm which doesn't use shortcut edges and always walks along back edges all the way to the root. Show an example of an input where this algorithm is asymptotically slower than the actual AC algorithm.

Frequencies of Occurrences. Describe an algorithm which, in time $\mathcal{O}(|T| + \sum_i |P_i|)$, outputs an array of frequencies, i.e. f_i is how many times P_i appears in T . (Notice that the complexity should not depend on the number of occurrences!)

2D Search. Given an $n \times n$ matrix A , decide whether it contains a given $m \times m$ submatrix B as a contiguous submatrix.

Fibonacci words. Let us define Fibonacci words as follows: $F_0 = \mathbf{a}$, $F_1 = \mathbf{b}$, $F_{n+2} = F_n F_{n+1}$. Design an algorithm which, in a given string over the alphabet $\{\mathbf{a}, \mathbf{b}\}$, finds the longest Fibonacci subword.

Dynamic Search. Design a data structure for dynamic searching. The pattern P is fixed, but the characters in the text T may change, and the data structure should always quickly answer whether the current text contains the pattern.

Shortest Word without Occurrence. Given a set of patterns P_1, \dots, P_n over the alphabet $\{\mathbf{a}, \dots, \mathbf{z}\}$ and a number $l \in \mathbb{N}$, compute the lexicographically smallest word of length l which does not contain any of the patterns.

d -dimensional search. Same as the task *2D search*, but generally in d dimensions.

Shortest Universal Word. Given are again patterns P_1, \dots, P_n over the alphabet $\{\mathbf{a}, \dots, \mathbf{z}\}$. Find the shortest word which contains all patterns. Aim for an algorithm with complexity $\mathcal{O}(2^n \sum_{i=1}^n |J_i|)$.